Scilab Textbook Companion for Data Structures Using C And C++ by Y. Langsam, M. Augenstein And A. M. Tenenbaum[1]

Created by
Dharmesh Majethiya
B.Tech (pursuing)
Computer Engineering
NIT Tiruchirappalli
College Teacher
Mr.Kunwar Singh
Cross-Checked by
Siddharth Jain

August 12, 2013

[1]Funded by a grant from the National Mission on Education through ICT, http://spoken-tutorial.org/NMEICT-Intro. This Textbook Companion and Scilab codes written in it can be downloaded from the "Textbook Companion Project" section at the website http://scilab.in

Book Description

Title: Data Structures Using C And C++

Author: Y. Langsam, M. Augenstein And A. M. Tenenbaum

Publisher: Prentice - Hall Of India Pvt. Ltd.

Edition: 2

Year: 2006

ISBN: 81-203-1177-9

Scilab numbering policy used in this document and the relation to the above book.

Exa Example (Solved example)

Eqn Equation (Particular equation of the above book)

AP Appendix to Example(Scilab Code that is an Appednix to a particular Example of the above book)

For example, Exa 3.51 means solved example 3.51 of this book. Sec 2.3 means a scilab code whose theory is explained in Section 2.3 of the book.

Contents

List of Scilab Codes

Chapter 1

Introduction To Data Structures

Scilab code Exa 1.1 To calcualte Average And Deviation

```
//Solved Example 1
//:To  calcualte Average And Deviation
function[avg]=average(a)
  i=1;
  [j,k]=size(a);
  j=0;
  for i=1:k
    j=j+a(i);
  end
  avg=j/k;
  dev=0;
  disp(avg,"Average =");
  disp("The deviations are:");
  for i=1:k
    dev=a(i)-avg;
    disp(dev);
  end
endfunction
//Calling routine
```

```
a=[3 223 212 343]
avg=average(a)
```

Scilab code Exa 1.1.4 Decimal form of given no represented variably

```
//Exercise1.1 Example.1.1.4
//To calculate  Decimal No. of a given Number
//Treating them as i)Normal binary nos(ii)Twos
   complemented iii)BCD:
function[c]=twos1(a1)
  [j1,i1]=size(a1)
  i4=1
  c=-(a1(i4)*2^(i1-1));
  i1=i1-1;
  while(i1>=1)
    i4=i4+1;
    c=c+a1(i4)*2^(i1-1);
    i1=i1-1;
  end
  disp(a1,"Decimal form of the Twos Complement
     Number");
  disp(c," is");
endfunction
function[d]=binary_dec(a2)
  [j2,i2]=size(a2);
  k=modulo(i2,4);
  d=0;
  if(k==0)
    e=i2/4;
    i3=1
    while(i3<=i2)
      l=3
      m=0
      while(l>=0)
        m=m+(a2(i3)*2^l);
```

```
        l=l-1;
        i3=i3+1;
      end
      if(m>9)
        d=-1;
        disp("Cannot be coded in this form")
       break;
      end
      if(m<=9)
        d=d+m*10^(e-1)
        e=e-1;
      end
    end
  end
  disp(a2,"Decimal form of  BCD number");
  disp(d,"is");
endfunction
//Given Example:
//(A)
p1=[1 0 0 1 1 0 0 1];
p2=base2dec(['10011001'],2)
p2=twos1(p1)
p2=binary_dec(p1)
//(b)
p3=[1 0 0 1];
p4=base2dec(['1001'],2)
p4=twos1(p3)
p4=binary_dec(p3)
//(C)
p5=[0 0 0 1 0 0 0 1 0 0 0 1];
p6=base2dec(['000100010001'],2)
p6=twos1(p5)
p6=binary_dec(p5)
//(d)
p7=[0 1 1 1 0 1 1 1];
p8=base2dec(['01110111'],2)
p8=twos1(p7)
p8=binary_dec(p7)
```

```
//(e)
p9=[0 1 0 1 0 1 0 1];
p10=base2dec(['01010101'],2)
p10=twos1(p9)
p10=binary_dec(p9)
//(F)
p11=[1 0 0 0 0 0 0 1 0 1 0 1];
p12=base2dec(['100000010101'],2)
p12=twos1(p11)
p12=binary_dec(p11)
```

Scilab code Exa 1.1.5 Add Substract And Multiply binary numbers

```
//Exercise 1.1 example 1.1.5
//Add,Substract And Multiply binary numbers
function[a]=add(b,c)
  d=base2dec(b,2)
  e=base2dec(c,2)
  a=d+e
  a=dec2bin(a)
  disp(a,"Result of addition")
endfunction
function[a]=subtract(b,c)
  d=base2dec(b,2)
  e=base2dec(c,2)
  a=d-e
  a=dec2bin(a)
  disp(a,"Result of subtraction")
endfunction
function[a]=multiply(b,c)
  d=base2dec(b,2)
  e=base2dec(c,2)
  a=d*e
  a=dec2bin(a)
  disp(a,"Result  of multiplication");
```

```
endfunction
//Calling Routine:
b="11001";
c="10011";
a=add(b,c)
a=subtract(b,c)
a=multiply(b,c)
```

Scilab code Exa 1.1.7 TO Convert Binary To Ternary

```
//Exercise 1.1 Example 1.1.7
//TO Convert Binary To Ternary
function[t]=bin_ter(a)
  b=0
  b=base2dec(a,2);
  disp(b);
  [j,i]=size(a);
  t=[];
  while(b~=0)
    m=modulo(b,3);
    t=[t(:,:) m];
    b=b/3;
    b=b-modulo(b,10);
  end
  disp(t,"Ternary Equivalent");
endfunction
//Calling Routine:
a="100101101110"
disp(a,"input string is");
b=bin_ter(a)
```

Scilab code Exa 1.2 String Manipulations

```
//Solved Example 2
//:String Manipulations
funcprot(0)
function[l]=strlen(str)
  i=1;
  l=0;
  [j,k]=size(str)
  for  i=1:k
    l=l+length(str(i));
  end
  disp(l,"string length is");
endfunction
//Calling Routine:
str="Hello World";
l=strlen(str)
function[c]=strcat1(a,b)
  disp(strcat([a b]),"After concatination");
  c=strcat([a b]);
endfunction
//Calling Routine:
a="hello ";
b="world";
c=strcat1(a,b);
```

Scilab code Exa 1.2.1 Calculate Median And Mode Of an Array

```
//Exercise Example 1.2.1
//Calculates Median And Mode Of an  Array
//(A)
function[y]=median1(a)
  p=mtlb_sort(a);
  [j,i]=size(a);
  y=0
  j=modulo(i,2);
  if(j==0)
```

```
    y=((a(i/2)+a(i/2+1))/2);
  end
  if(j==1)
    i=i/2;
    i=i-modulo(i,10);
    y=a(i+1);
  end
  disp(y,"median is");
endfunction
//(B)
function[z]=mode1(a)
  p=mtlb_sort(a);
  disp(p)
  q=1;
  r=1;
  i=1;
  [j,i1]=size(a);
  if(i1>1)
    for i=1:i1-1
      if(p(i)~=p(i+1))
      q=[q(:,:) i+1];
      r=[r(:,:) 1];
      else
        [c,d]=size(r);
        r(d)=r(d)+1;
      end
    end
    q1=mtlb_sort(r);
    [j,i1]=size(q1)
    if(q1(i1-1)==q1(i1))
      z=-1;
      disp("Mode does not exist");
      break;
      else
        c=q1(i1);
        k=1;
        while(r(k)~=c)
          k=k+1;
```

```
        end
        z=p(q(k));
      end
    end
    if(i1==1)
      z=a(1);
    end
    disp(z,"mode is");
endfunction
a=[223 12 233322 121]
y=median1(a);
z=mode1(a);
```

Scilab code Exa 1.2.6 Finding the adress in a row major array

```
//Exercise1.2 Example 1.2.6
//Finding the adress in a row major array
function[]=add(m,n)
  printf("Adress is %d\n",m+n*20);
endfunction

//(a)
add(10,0);
//(b)
add(100,0);
//(c)
add(0,0);
//(d)
add(2,1);
//(e)
add(5,1);
//(f)
add(1,10);
//(g)
add(2,10);
```

```
//(h)
add(5,3);
//(i)
add(9,19);
```

Scilab code Exa 1.3 Writing name from structure and counting alphabets

```
//Solved Example 5:
//Writing a name from the given structure and
//counting the number of alphabets printed
function[l]=strlen(str)
  i=1;
  l=0;
  [j,k]=size(str)
  for  i=1:k
    l=l+length(str(i));
  end
endfunction
function[count]=writename(name)
  printf("\n");
  printf("%s",name.first);
  printf("%c",'  ');
  printf("%s",name.midinit);
  printf("\t");
  printf("%s",name.last);
  printf("\n");

  a=string(name.first);
  count=strlen(a);
  a=string(name.midinit);
  count=count+strlen(a);
  a=string(name.last);
  count=count+strlen(a);
  disp(count,"Count is:");
endfunction
```

```
//Calling Routine
name=struct('first','praveen','midinit','rajeev','last','chauhan');
count=writename(name)
```

Scilab code Exa 1.3.1 Implementing Complex Numbers by structure

```
//Exercise 1.3
//Example 1.3.1
//Implementing Complex Numbers by structure
function[]=complexmanu(x1,x2,x3,x4)

  com1=struct('real',x1,'complex',x2);
  com2=struct('real',x3,'complex',x4);
   //adding 2 numbers
   add=struct('real',x1+x3,'complex',x2+x4);
   disp(add.complex,"+ i",add.real,"Addition result is ");
   //Substract
   sub=struct('real',x1-x3,'complex',x2-x4);
   disp(sub.complex,"+ i",sub.real,"Substraction result is ");
   //Negating
   neg=struct('real',-x1,'complex',-x2);
   disp(neg.complex,"+ i",neg.real,"Negation result for the first is ");
   //Multiplication
   mul=struct('real',x1*x3-x2*x4,'complex',x2*x3+x4*x1);
    disp(mul.complex,"+ i",mul.real,"Multiplication result is ");
 endfunction
 x1=3;
 x2=5;
 x3=5;
```

```
x4=6;
complexmanu(x1,x2,x3,x4);
```

Scilab code Exa 1.3.6 Adding Substracting and multiplying Rational Nos

```
//Exercise 1.3
//Example 1.3.6
//Adding,Subtracting and multiplying Rational
   Numbers
function[]=rational(x1,x2,x3,x4)
rational1=struct('numerator',x1,'denominator',x2);
disp(rational1);
rational2=struct('numerator',x3,'denominator',x4);
disp(rational2);
//Add
x5=int32([x2 x4]);
x5=lcm(x5);
x6=x1*(x5/x2)+x3*(x5/x4);
rational3=struct('numerator',x6,'denominator',x5);
disp(rational3,"After addition");
//subtract
x6=x1*(x5/x2)-x3*(x5/x4)
rational4=struct('numerator',x6,'denominator',x5);
disp(rational4,"After Subtraction");
//Multiply
x7=x1*x3;
x8=x2*x4;
rational5=struct('numerator',x7,'denominator',x8);
disp(rational5,"After multiplication");
endfunction
x1=43;
x2=32;
x3=233;
x4=33;
rational(x1,x2,x3,x4);
```

Scilab code Exa 1.3.7 Checking Equality Of 2 Rational Numbers

```
//Exercise 1.3
//Example 1.3.7
//Checking Equality Of 2 Rational Numbers Without
   Reducing Them
function []=rational_equal(x1,x2,x3,x4)
rational1=struct('numerator',x1,'denominator',x2);
disp(rational1);
rational2=struct('numerator',x3,'denominator',x4);
disp(rational2);
if(x1*x4==x2*x3)
  disp("Equal");
  break;
else
  disp("Not  Equal");
  break;
end
endfunction
//Calling Routine:
x1=32;
x2=45;
x3=43;
x4=55;
rational_equal(x1,x2,x3,x4);
```

Scilab code Exa 1.4 Raising the salary of employee

```
//Solved Example 6
//To Raise The salary of an employee
function[employee1]=raise(employee,n)//employee is
   the list of employees
```

```
  for i=1:n
    if(employee(i)(1).year<=2000)
      employee(i)(2)=employee(i)(2)*1.1;
    else
      employee(i)(2)=employee(i)(2)*1.05;
    end
  end
  employee1=employee;
  disp("After Raising");
  for i=1:n
    printf("Employee no %d\n",i);
    disp(employee(i)(1));
    disp(employee(i)(2));
  end

endfunction
//Calling Routine:
datehired=struct('year',1993,'month',12);
employee1=list(datehired,14000);
datehired=struct('year',1998,'month',12);
employee2=list(datehired,17000);
datehired=struct('year',2003,'month',12);
employee3=list(datehired,25000);
datehired=struct('year',2002,'month',12);
employee4=list(datehired,35000);
datehired=struct('year',2006,'month',12);
employee5=list(datehired,13000);
employee=list(employee1,employee2,employee3,
   employee4,employee5);
employee=raise(employee,5)
```

Scilab code Exa 1.5 Reducing the given rational number

```
//Solved Example 7:
//Reducing The Given Rational Number
```

```
funcprot(0)
function[y]=reduce(nm,dn)
rational1=struct('numerator',nm,'denominator',dn)
y=0
if(rational1.numerator>rational1.denominator)
  a=rational1.numerator;
  b=rational1.denominator;
else
  a=rational1.denominator;
  b=rational1.numerator;
end
while(b~=0)
  rem=modulo(a,b);
  a=b;
  b=rem;
end
y=struct('numerator',nm/a,'denominator',dn/a);
disp(y);
endfunction
nm=22;
dn=44;
y=reduce(nm,dn)
```

Scilab code Exa 1.6 Equality check of 2 rational nos by reduction

```
//Solved Example 8:
//Checking for the equality of 2 rational numbers by
    reducing them
function[]=equal(x1,x2,x3,x4)
  rational1=struct('numerator',x1,'denominator',x2)
 rational2=struct('numerator',x3,'denominator',x4)
 y=0
if(rational1.numerator>rational1.denominator)
  a=rational1.numerator;
  b=rational1.denominator;
```

```
else
  a=rational1.denominator;
  b=rational1.numerator;
end
while(b~=0)
  rem=modulo(a,b);
  a=b;
  b=rem;
end
y=struct('numerator',x1/a,'denominator',x2/a);
y1=0
if(rational2.numerator>rational2.denominator)
  a=rational2.numerator;
  b=rational2.denominator;
else
  a=rational2.denominator;
  b=rational2.numerator;
end
while(b~=0)
  rem=modulo(a,b);
  a=b;
  b=rem;
end
y1=struct('numerator',x3/a,'denominator',x4/a);
if(y==y1)
  disp("Equal")
  break;
else
  disp("Not Equal")
  break;
end
endfunction
x1=5;
x2=7;
x3=35;
x4=49;
equal(x1,x2,x3,x4);
```

Chapter 2

Stacks

Scilab code Exa 2.1 To determine the syntacticaly valid string

```
//Solved Example 1
//To determine the syntacticaly valid string
function[l]=strlen(x)
  i=1;
  l=0;
  [j,k]=size(x)
  for i=1:k
    l=l+length(x(i));
  end
endfunction
function[]=stringvalid(str)
  str=string(str);
  stack=struct('a','0','top',0);
  l1=strlen(str);
  valid=1;
  l=1;
  while(l<=l1)
      if(str(l)=='('|str(l)=='['|str(l)=='{')
        if(stack.top==0)
          stack.a=str(l);
          stack.top=stack.top+1;
```

```
      else
        stack.a=[stack.a(:,:) str(l)];
        stack.top=stack.top+1;
      end
    end
    if(str(l)==')'|str(l)==']'|str(l)=='}')
      if(stack.top==0)
        valid=0;
        break;
      else
       i=stack.a(stack.top);
       stack.top=stack.top-1;
       symb=str(l);
       if(((symb==')')&(i=='('))|((symb==']')&(i==
          '['))|((symb=='}')&(i=='{')))
      else
        valid=0;
        break;
      end
    end
  end
    l=l+1;
  end
  if(stack.top~=0)
    valid=0;
  end
  if(valid==0)
    disp("Invalid String");
  else
    disp("Valid String");
  end
endfunction
//Calling Routine:
stringvalid(['H' 'E' 'L' 'L' 'O'])
```

Scilab code Exa 2.1.2 To determine the syntacticaly valid string

```
//Solved Example 1
//To  determine the syntacticaly valid string
function[l]=strlen(x)
  i=1;
  l=0;
  [j,k]=size(x)
  for  i=1:k
    l=l+length(x(i));
  end
endfunction
function[]=stringvalid(str)
  str=string(str);
  stack=struct('a','0','top',0);
  l1=strlen(str);
  valid=1;
  l=1;
  while(l<=l1)
      if(str(l)=='('|str(l)=='['|str(l)=='{')
        if(stack.top==0)
          stack.a=str(l);
          stack.top=stack.top+1;
        else
          stack.a=[stack.a(:,:) str(l)];
          stack.top=stack.top+1;
        end
        disp(stack);
      end
      if(str(l)==')'|str(l)==']'|str(l)=='}')
        if(stack.top==0)
          valid=0;
          break;
        else
          i=stack.a(stack.top);
          b=stack.a(1);
          for i1=2:stack.top-1
            b=[b(:,:) stack.a(i1)]
```

```
          end
          stack.a=b;
         stack.top=stack.top-1;
         symb=str(l);
         disp(stack);
         if(((symb==')')&(i=='('))|((symb==']')&(i='['))|((symb='}')&(i='{')))
        else
          valid=0;
          break;
        end
      end
    end
      l=l+1;
    end
    if(stack.top~=0)
      valid=0;
    end
    if(valid==0)
      disp("Invalid String");
    else
      disp("Valid String");
    end
  endfunction
  //Calling Routine:
  stringvalid(['(' 'A' '+' 'B' '}' ')'])
  stringvalid(['{' '[' 'A' '+' 'B' ']' '-' '[' '(' 'C' '-' 'D' ')' ']'])
  stringvalid(['(' 'A' '+' 'B' ')' '-' '{' 'C' '+' 'D' '}' '-' '[' 'F' '+' 'G' ']'])
  stringvalid(['(' '(' 'H' ')' '*' '{' '(' '[' 'J' '+' 'K' ']' ')' '}' ')'])
  stringvalid(['(' '(' '(' 'A' ')' ')' ')'])
```

Scilab code Exa 2.2 Implementing Stack using union

```
//Solved Example 2:
//Implementing Stack using union:
function[stack]=sta_union(etype,a)
  stackelement=struct('etype',etype);
  [k,l]=size(a);
select stackelement.etype,
case 'int' then
  a=int32(a);
  stack=struct('top',l,'items',a);,
  case 'float' then
  a=double(a);
  stack=struct('top',l,'items',a);,
  case 'char' then
  a=string(a);
  stack=struct('top',l,'items',a);,
end
disp(stack,"Stack is:");
endfunction
a=[32 12.34 232 32.322]
stack=sta_union('float',a)
stack=sta_union('int',a)
stack=sta_union('char',a)
```

Scilab code Exa 2.2.3 Check if string is of certain form

```
function[l]=strlen(x)
  i=1;
  l=0;
  [j,k]=size(x)
  for i=1:k
    l=l+length(x(i));
  end
endfunction
function[]=str(st)
  stack=struct('a',0,'top',0);
```

```
st=string(st);
l=1;
l1=strlen(st);
symb=st(l);
valid=1;
while(l<l1)
  while(symb~='C')
    if(stack.top==0)
        stack.a=st(l);
        stack.top=stack.top+1;
     else
        stack.a=[stack.a(:,:) st(l)];
        stack.top=stack.top+1;
     end
     l=l+1;
     symb=st(l);
   end
   i=st(l+1);
  if(stack.top==0)
      valid=0;
      break;
   else
     symb1=stack.a(stack.top);
     stack.top=stack.top-1;
     if(i~=symb1)
       valid=0;
       break;
     end
   end
  l=l+1;
end
  if(stack.top~=0)
    valid=0;
  end
  if(valid==0)
    disp("Not of the given format");
  else
    disp("String Of the Given Format");
```

```
    end
endfunction
//Calling Routine:
st=['A' 'A' 'B' 'A' 'C' 'A' 'B' 'A' 'A']
str(st)
st=['A' 'A' 'B' 'A' 'C' 'A' 'B' 'A' ]
str(st)
```

Scilab code Exa 2.3 Implementing Push And Pop Functions

```
//Solved Example 3:
//Implementing Push And Pop Functions:
function[y,sta1]=empty(sta)
  y=0;
  sta1=0;
  if(sta.top==0)
    y=0;
  else
    y=1;
  end
  sta1=sta
endfunction

function[sta]=push(stac,ele)
  sta=0;
  if(empty(stac)==0)
    stac.a=ele;
    stac.top=stac.top+1;
  else
    stac.a=[stac.a(:,:) ele]
    stac.top=stac.top+1;
  end
  disp(stac);
  sta=stac;
endfunction
```

```
function[ele,sta]=pop(stack)
  ele='-1';
  if(empty(stack)==0)
    disp("Stack Underflow");
    break;
  else
    ele=stack.a(stack.top);
    stack.top=stack.top-1;
    if(stack.top~=0)
      b=stack.a(1);
    for i2=2:stack.top
      b=[b(:,:) stack.a(i2)];
    end
    stack.a=b;
  else
    stack.a='0';
  end
  end
  disp(stack);
  sta=stack;
endfunction
global stack
//Calling Routine:
stack=struct('a',0,'top',0);
stack=push(stack,4);
stack=push(stack,55);
stack=push(stack,199);
stack=push(stack,363);
[ele,stack]=pop(stack);
disp(stack,"After the above operations stack is:");
```

Scilab code Exa 2.4 Convering an infix expression to a Postfix Express

```
//Solved Example 5:
```

```
//Convering an infix expression to a Postfix
    Expression:
function[sta]=push(stac,ele)
  sta=0;
  if(stac.top==0)
    stac.a=ele;
    stac.top=stac.top+1;
  else
    stac.a=[stac.a(:,:) ele]
    stac.top=stac.top+1;
  end
  disp(stac);
  sta=stac;
endfunction

function[ele,sta]=pop(stack)
  ele='-1';
  if(stack.top==0)
    disp("Stack Underflow");
    break;
  else
    ele=stack.a(stack.top);
    stack.top=stack.top-1;
    if(stack.top~=0)
      b=stack.a(1);
    for i2=2:stack.top
      b=[b(:,:) stack.a(i2)];
    end
    stack.a=b;
  else
    stack.a='0';
  end
  end
  sta=stack;
endfunction
function[l]=strlen(x)
  i=1;
  l=0;
```

```
  [j,k]=size(x)
  for  i=1:k
    l=l+length(x(i));
  end
endfunction
function[p]=pre(s1,s2)
     i1=0;
    select s1,
    case '+' then i1=5;
    case '-' then i1=5;
    case '*' then  i1=9;
    case '/' then i1=9;
    end
    i2=0;
    select s2,
    case '+' then i2=5;
    case '-' then i2=5;
    case '*' then  i2=9;
    case '/' then i2=9;
    end
    p=0;
    p=i1-i2;
    if(s1=='(')
      p=-1;
    end
    if(s2=='('&s1~=')')
      p=-1;
    end
    if(s1~='('&s2==')')
      p=1;
    end

  endfunction
function[a2]=intopo(a1,n)
  stack=struct('a',0,'top',0);
  l1=1;
  l2=strlen(a1(1))
  for i=2:n
```

```
    l2=l2+strlen(a1(i))
  end
  a2=list();
  while(l1<=l2)
    symb=a1(l1);
    if(isalphanum(string(a1(l1))))
      a2=list(a2,symb);
    else
      while(stack.top~=0&(pre(stack.a(stack.top),
         symb)>=0))
        [topsymb,stack]=pop(stack);
        if(topsymb==')'|topsymb=='(')
          a2=a2;
        else
          a2=list(a2,topsymb);
        end
      end
      if(stack.top==0|symb~=')')
        stack=push(stack,symb);
      else
        [ele,stack]=pop(stack);
      end
    end
    l1=l1+1;
  end
  while(stack.top~=0)
    [topsymb,stack]=pop(stack);
    if(topsymb==')'|topsymb=='(')
      a2=a2;
    else
      a2=list(a2,topsymb);
    end
  end
  disp(a2);
endfunction
//Calling Routine:
a1=['(' '2' '+' '3' ')' '*' '(' '5' '-' '4' ')']
a2=intopo(a1,11)
```

Chapter 3

Recursion

Scilab code Exa 3.1 Multiplication of 2 numbers

```
//Multiplication of 2 numbers
funcprot(0)
function[val]=mul(a,b)
  if(b==1)
    val=a;
  else
    val=a+mul(a,b-1);
  end
endfunction
//Calling Routine:
a=4;
b=3;
val=mul(4,3)
printf("Product of %d and %d is %d",a,b,val);
```

Scilab code Exa 3.2 Factorial of a number

```
//Function To Caluculate factorial of a given
   number
```

```
function[value]=fact(a)
  value=-1;
  if(a<0|a>170)
    disp("Invalid valu.");
    break;
  else
    if(a==1|a==0)
      value=1;
    else
      value=a*fact(a-1);
    end
  end
endfunction
//Calling Routine:
a=5;
val=fact(a);
printf("%d factorial is %d",a,val);
```

Scilab code Exa 3.3 Fibbonacci series

```
function[fib]=fibbo(n)
  fib=-1;
  if(n<0)
    disp("Invalid Entry");
  else
   if(n<=1)
     fib=n;
   else
     fib=fibbo(n-1)+fibbo(n-2);
   end
 end
endfunction

function[l]=fibbon(n)
  x=0;
```

```
  l=(fibbo(0));
  for x=1:n-1
   l=[l(:,:),fibbo(x)];
  end
 disp(l);
endfunction
//Calling Routine:
n=5;
l=fibbon(n)
```

Scilab code Exa 3.4 Binary Search

```
function[b]=bsear(a,l,u,n)
  if(l>u)
    b=-1;
  else
    mid=int32((l+u)/2);
    if(n==a(mid))
      b=n;
    else
      if(n>a(mid))
         mid=int32((l+u)/2);
        b=bsear(a,mid+1,u,n);
      else
         mid=int32((l+u)/2);
        b=bsear(a,l,mid-1,n);
      end
    end
  end
endfunction

function[b]=bsearc(a,l,u,n)
  b=bsear(a,l,u,n);
  if(b==-1)
    disp("The element is not there");
```

```
  end
  if(b==n)
    disp("The element is there");
  end
endfunction
//Calling Routine:
a=[12 122 3233 12121]//Must be sorted:
b=bsearc(a,1,4,12)
```

Scilab code Exa 3.5 Tower Of Hanoi

```
function []=towe(n,from,to,aux)
  if(n==1);
    disp(to,"to ",from,"Move peg 1 from");
  else
    towe(n-1,from,aux,to);
    disp(to,"to",from,"from",n,"Move Peg");
    towe(n-1,aux,to,from);
  end
endfunction

function []=tower(from,to,aux)
  n=input("Enter n");
  towe(n,from,to,aux);
endfunction
//Calling Routine:
n=3//Number of disks
towe(n,'a','b','c')
```

Scilab code Exa 3.6 Prefix To Postfix Conversion

```
funcprot(0)
function[y]=find1(g)
```

```
  length1=strlen(g);
  if(length1==0)
    y=0;
  else
    if(isalphanum(g(1)))
      y=1;
    else
      if(length1<2)
        y=0;
      else
         s=strsplit(g,1);
         s=s(2);
         m=find1(s);
         if(m==0|length1==m)
           y=0;
         else
            e=strsplit(g,m+1);
            e=e(2);
            n=find1(e);
            if(n==0)
              y=0;
            else
              y=m+n+1;
            end
          end
        end
      end
    end
  endfunction
  function[l]=strlen(x)
  i=1;
  l=0;
  [j,k]=size(x)
  for  i=1:k
    l=l+length(x(i));
  end
endfunction
function[po]=pr2po(pr)
```

```
 length1=strlen(pr);
 if(length1==1)
   if(isalphanum(pr))
     po(1)=pr(1);
   else
     disp("Invalid string\n");
   end
 else
   s=strsplit(pr,1);
   g=s(2);
   m=find1(g);
   s=strsplit(pr,m+1);
   g1=s(2);
   n=find1(g1);
   f=strsplit(pr,1);
   c=f(1);
   if((c~='+'&c~='-'&c~='/'&c~='*')|m==0|n==0|m+n+1~=length1)
     printf("Invalid string\n");
   else
     s=strsplit(pr,1);
     s=strsplit(s(2),m);
     opnd1=s(1);
     s=strsplit(pr,m+1);
     opnd2=s(2);
     post1=pr2po(opnd1);
     post2=pr2po(opnd2);
     post=[post1(:,:) post2(:,:)]
     f=strsplit(pr,1);
     c=f(1);
     post3=[post(:,:) c];
     po=post3;
    end
  end
endfunction
//Calling Routine:

s1="+-*abcd";//no spaces between
```

```
po=pr2po(s1);
disp(po,"postfix is");
s1="+-*/+-*/abcdefghi"
po=pr2po(s1);
disp(po,"postfix is");
```

Scilab code Exa 3.7 Simulating Factorial By Non recursion

```

function []=simu_fact(n);
  a=1;
  while(n>0)
    a=a*n;
    n=n-1;
  end
  disp(a,"Factorial is ");
endfunction
//Calling Routine:
a=9
simu_fact(a)
```

Chapter 4

Queues and linked list

Scilab code Exa 4.1 Implementing Singly Connected Linked List

```
//SINGLY CONNECTED LINKED LIST:
function[link2]=append(ele,link1)
  link2=list(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,,0,0);
  if(link1(1)(1).add==0)
    link1(1)(1).data=ele;
    link1(1)(1).add=1;
    link1(1)(1).nexadd=0;
    link2(1)=link1(1)(1);
    else
    if(link1(1)(1).nexadd==0)
      lin2=link1(1)(1);
      lin2.data=ele;
      lin2.add=link1(1)(1).add+1;
      link1(1)(1).nexadd=lin2.add;
      lin2.nexadd=0;
      link2(1)=link1(1)(1);
      link2(2)=lin2;
    else
      lin2=link1(1)(1);
```

```
      i=1;
      while(link1(i)(1).nexadd~=0)
        i=i+1;
      end
      j=i;
      lin2.data=ele;
      lin2.add=link1(i).add+1;
      lin2.nexadd=0;
      link1(i).nexadd=lin2.add;
      link2(1)=link1(1)(1);
      i=2;
      while(link1(i).nexadd~=lin2.add)
        link2(i)=(link1(i));
        i=i+1;
      end
      link2(i)=link1(i);
      link2(i+1)=lin2;
    end
  end
endfunction
function[link2]=add(ele,pos,link1);
   link2=list(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,,0,0);
   i=1;
   while(i<=pos)
     if(link1(i).nexadd==0)
       break;
     else
       i=i+1;
     end
   end
   if(link1(i).nexadd~=0)
     i=i-1;
      lin2.data=ele;
      lin2.add=i;
      j=i;
      while(link1(j).nexadd~=0)
```

```
          link1(j).add=link1(j).add+1;
          link1(j).nexadd=link1(j).nexadd+1;
          j=j+1;
        end
        link1(j).add=link1(j).add+1;
        lin2.nexadd=link1(i).add;
        link1(i-1).nexadd=lin2.add;
        k=1;
        while(k<i)
           link2(k)=link1(k);
           k=k+1;
         end
         link2(k)=lin2;
         k=k+1;
         link2(k)=link1(k-1);
         k=k+1
         l=k-1;
         while(k~=j)
           link2(k)=link1(l);
           k=k+1;
           l=l+1;
         end
         link2(j)=link1(j-1);;
         link2(j+1)=link1(j);
       else
         if(i==pos&i~=1)
           k=1;
           lin2.data=ele;
           lin2.add=link1(i-1).add+1;
           link1(i).add=link1(i).add+1;
           lin2.nexadd=link1(i).add;
           k=1;
           while(k<pos)
             link2(k)=link1(k);
             k=k+1;
           end
           link2(k)=lin2;
           link2(k+1)=link1(k)
```

```
        end
        if(i==pos&i==1)
          link2=append(ele,link1);
          return link2;
        end
      end
endfunction
function[link2]=delete1(pos,link1)
  link2=list
     (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,,0,0)
     ;
  i=1;
  while(i<=pos)
    if((link1(i).nexadd==0))
      break;
    else
      i=i+1;
    end
  end
  if(link1(i).nexadd~=0)
    i=i-1;
    j=1;
    if(i==1)
      j=1;
      while(link1(j).nexadd~=0)
        link2(j)=link1(j);
        j=j+1;
      end
      link2(j)=link1(j);
    else
    link1(i-1).nexadd=link1(i+1).add;
    while(link1(j).nexadd~=link1(i+1).add)
      link2(j)=link1(j);
      j=j+1;
    end
    if(j~=i-1)
      link2(j)=link1(j);
      link2(j+1)=link1(j+1);
```

```
      k=i+1;
      l=2;
    else
      link2(j)=link1(j);
      k=i+1;
      l=1;
    end
    while(link1(k).nexadd~=0)
      link2(j+l)=link1(k);
      k=k+1;
      l=l+1;
    end
    link2(j+l)=link1(k);
  end
  else
    if(i==pos)
      j=1;
      link1(i-1).nexadd=0;
      while(j<=i-1)
        link2(j)=link1(j);
        j=j+1;
      end
    end
  end
endfunction



//Calling Routine:
link1=struct('data',0,'add',0,'nexadd',0);//Creates
   empty list
link1=append(4,link1)
link1=append(6,link1)
link1=add(7,2,link1)
link1=append(8,link1)
link1=delete1(4,link1)
disp(link1,"The linked list after the above
   modifications is:");
```

Scilab code Exa 4.2 Implementing Queue Operarions

```
//Queue  Operations
function[q2]=push(ele,q1)
  if(q1.rear==q1.front)
    q1.a=ele;
    q1.rear=q1.rear+1;
  else
    q1.a=[q1.a(:,:) ele];
    q1.rear=q1.rear+1;
  end
  q2=q1;
endfunction
function[ele,q2]=pop(q1)
  ele=-1;
  q2=0;
  if(q1.rear==q1.front)
    disp("Queue  Underflow");
    return;
  else
    ele=q1.a(q1.rear-q1.front);
    q1.front=q1.front+1;
    i=1;
    a=q1.a(1);
    for i=2:(q1.rear-q1.front)
      a=[a(:,:) q1.a(i)];
    end
    q1.a=a;
  end
  q2=q1;
endfunction
//Calling  Routine:
q1=struct('a',0,'rear',0,'front',0)
q1=push(3,q1)
```

```
q1=push(22,q1);
q1=push(21,q1);
disp(q1,"Queue after insertion");
[ele,q1]=pop(q1)
disp(ele,"poped element");
disp(q1,"Queue after poping");
[ele,q1]=pop(q1);
[ele,q1]=pop(q1);
[ele,q1]=pop(q1);//Underflow Condition
```

Scilab code Exa 4.3 Implementing Circular Linked List

```
//CIRCULAR LINKED LIST
function[link2]=append(ele,link1)
  link2=list(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,,0,0);
  if(link1(1)(1).add==0)
    link1(1)(1).data=ele;
    link1(1)(1).add=1;
    link1(1)(1).nexadd=1;
    link2(1)=link1(1)(1);
    else
    if(link1(1)(1).nexadd==link1(1)(1).add)
      lin2=link1(1)(1);
      lin2.data=ele;
      lin2.add=link1(1)(1).add+1;
      link1(1)(1).nexadd=lin2.add;
      lin2.nexadd=link1(1)(1).add;
      link2(1)=link1(1)(1);
      link2(2)=lin2;
    else
      lin2=link1(1)(1);
      i=1;
      while(link1(i)(1).nexadd~=link1(1)(1).add)
```

```
        i=i+1;
      end
      j=i;
      lin2.data=ele;
      lin2.add=link1(i).add+1;
      lin2.nexadd=link1(1)(1).add;
      link1(i).nexadd=lin2.add;
      link2(1)=link1(1)(1);
      i=2;
      while(link1(i).nexadd~=lin2.add)
        link2(i)=(link1(i));
        i=i+1;
      end
      link2(i)=link1(i);
      link2(i+1)=lin2;
    end
  end
endfunction
function[link2]=add(ele,pos,link1);
   link2=list
      (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,,0,0)
      ;
   i=1;
   while(i<=pos)
     if(link1(i).nexadd==link1(1)(1).add)
       break;
     else
       i=i+1;
     end
   end
   if(link1(i).nexadd~=link1(1)(1).add)
     i=i-1;
      lin2.data=ele;
      lin2.add=i;
      j=i;
      while(link1(j).nexadd~=link1(1)(1).add)
        link1(j).add=link1(j).add+1;
        link1(j).nexadd=link1(j).nexadd+1;
```

```
      j=j+1;
    end
    link1(j).add=link1(j).add+1;
    lin2.nexadd=link1(i).add;
    link1(i-1).nexadd=lin2.add;
    k=1;
    while(k<i)
       link2(k)=link1(k);
       k=k+1;
     end
     link2(k)=lin2;
     k=k+1;
     link2(k)=link1(k-1);
     k=k+1
     l=k-1;
     while(k~=j)
       link2(k)=link1(l);
       k=k+1;
       l=l+1;
     end
     link2(j)=link1(j-1);;
     link2(j+1)=link1(j);
   else
     if(i==pos)
       k=1;
       lin2.data=ele;
       lin2.add=link1(i-1).add+1;
       link1(i).add=link1(i).add+1;
       lin2.nexadd=link1(i).add;
       link1(i).nexadd=link1(1)(1).add;
       k=1;
       while(k<pos)
         link2(k)=link1(k);
         k=k+1;
       end
       link2(k)=lin2;
       link2(k+1)=link1(k)
     end
```

```
    end

endfunction
function[link2]=delete1(pos,link1)
  link2=list(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,,0,0);
  i=1;
  j=1;
  while(i<pos)
    if((link1(j).nexadd==link1(1)(1).add))
      j=1;
      i=i+1;
    else
      i=i+1;
      j=j+1;
    end
  end
  if(link1(j).nexadd~=link1(1)(1).add)
    k=1;
    if(j==1)
      k=2;
      while(link1(k).nexadd~=link1(1)(1).add)
        link2(k-1)=link1(k);
        k=k+1;
      end
      link2(k-1)=link1(k);
      link2(k-1).nexadd=link2(1).add;
    else
      lin2=link1(j);
      link1(j-1).nexadd=link1(j+1).add;
      k=1;
      while(link1(k).nexadd~=link1(j+1).add)
        link2(k)=link1(k);
        k=k+1;
      end
      link2(k)=link1(k);
      k=k+2;
```

```
      while(link1(k).nexadd~=link1(1)(1).add)
        link2(k-1)=link1(k);
        k=k+1;
      end
      link2(k-1)=link1(k);
    end
  else
    link1(j-1).nexadd=link1(1)(1).add;
    l=1;
    while(link1(l).nexadd~=link1(1)(1).add)
      link2(l)=link1(l);
      l=l+1;
    end
    link2(l)=link1(l);
  end
endfunction
//Calling Routine:
link1=struct('data',0,'add',0,'nexadd',0);
link1=append(4,link1);//This will actualy create a
   list and 4 as start
link1=append(6,link1);
link1=add(10,2,link1);
link1=delete1(4,link1);//As the list is circular the
    4'th element refers to actualy the 1'st one
disp(link1,"After the above manuplations the list is
   ");
```

Scilab code Exa 4.4 Implementing Doubly connected Linked List

```
//DOUBLE LINKED LIST:
function[link2]=append(ele,link1)
  link2=list
     (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,,0,0)
     ;
  if(link1(1)(1).add==0)
```

```
    link1(1)(1).data=ele;
    link1(1)(1).add=1;
    link1(1)(1).nexadd=0;
    link1(1)(1).prevadd=0;
    link2(1)=link1(1)(1);
    else
    if(link1(1)(1).nexadd==0)
      lin2=link1(1)(1);
      lin2.data=ele;
      lin2.add=link1(1)(1).add+1;
      link1(1)(1).nexadd=lin2.add;
      lin2.nexadd=0;
      lin2.prevadd=link1(1)(1).add;
      link2(1)=link1(1)(1);
      link2(2)=lin2;
    else
      lin2=link1(1)(1);
      i=1;
      while(link1(i)(1).nexadd~=0)
        i=i+1;
      end
      j=i;
      lin2.data=ele;
      lin2.add=link1(i).add+1;
      lin2.nexadd=0;
      link1(i).nexadd=lin2.add;
      lin2.prevadd=link1(i).add;
      link2(1)=link1(1)(1);
      i=2;
      while(link1(i).nexadd~=lin2.add)
        link2(i)=(link1(i));
        i=i+1;
      end
      link2(i)=link1(i);
      link2(i+1)=lin2;
    end
  end
endfunction
```

```
function[link2]=add(ele,pos,link1);
  link2=list
     (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,,0,0)
     ;
  i=1;
  while(i<=pos)
    if(link1(i).nexadd==0)
      break;
    else
      i=i+1;
    end
  end
  if(link1(i).nexadd~=0)
    i=i-1;
     lin2.data=ele;
     lin2.add=i;
     j=i;
     while(link1(j).nexadd~=0)
       link1(j).prevadd=link1(j).prevadd+1;
       link1(j).add=link1(j).add+1;
       link1(j).nexadd=link1(j).nexadd+1;
       j=j+1;
     end
     link1(j).prevadd=link1(j).prevadd+1;
     link1(j).add=link1(j).add+1;
     lin2.nexadd=link1(i).add;
     link1(i).prevadd=lin2.add;
     lin2.prevadd=link1(i-1).add;
     link1(i-1).nexadd=lin2.add;
     k=1;
     while(k<i)
        link2(k)=link1(k);
        k=k+1;
      end
      link2(k)=lin2;
      k=k+1;
      link2(k)=link1(k-1);
      k=k+1
```

```
      l=k-1;
      while(k~=j)
        link2(k)=link1(l);
        k=k+1;
        l=l+1;
      end
      link2(j)=link1(j-1);;
      link2(j+1)=link1(j);
    else
      if(i==pos)
        k=1;
        lin2.data=ele;
        lin2.add=link1(i-1).add+1;
        link1(i).add=link1(i).add+1;
        lin2.nexadd=link1(i).add;
        link1(i).prevadd=lin2.add;
        lin2.prevadd=link1(i-1).add;
        k=1;
        while(k<pos)
          link2(k)=link1(k);
          k=k+1;
        end
        link2(k)=lin2;
        link2(k+1)=link1(k)
      end
    end

endfunction
function[link2]=delete1(pos,link1)
  link2=list(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,,0,0);
  i=1;
  while(i<=pos)
    if((link1(i).nexadd==0))
      break;
    else
      i=i+1;
```

```
      end
    end
    if(link1(i).nexadd~=0)
      i=i-1;
      j=1;
      if(i==1)
        j=1;
        while(link1(j).nexadd~=0)
          link2(j)=link1(j);
          j=j+1;
        end
        link2(j)=link1(j);
      else
        link1(i-1).nexadd=link1(i+1).add;
        link1(i+1).prevadd=link1(i-1).add;
      while(link1(j).nexadd~=link1(i+1).add)
        link2(j)=link1(j);
        j=j+1;
      end
      if(j~=i-1)
        link2(j)=link1(j);
        link2(j+1)=link1(j+1);
        k=i+1;
        l=2;
      else
        link2(j)=link1(j);
        k=i+1;
        l=1;
      end
      while(link1(k).nexadd~=0)
        link2(j+l)=link1(k);
        k=k+1;
        l=l+1;
      end
      link2(j+l)=link1(k);
    end
    else
      if(i==pos)
```

```
      j=1;
      link1(i-1).nexadd=0;
      while(j<=i-1)
        link2(j)=link1(j);
        j=j+1;
      end
    end
  end
endfunction
//Calling Routine:
link1=struct('data',0,'add',0,'nexadd',0);
link1=append(4,link1);
link1=append(6,link1);
link1=add(10,2,link1);
link1=delete1(3,link1);
disp(link1,"After the above manuplations the list is
   ");
```

Scilab code Exa 4.5 Implementing Stack using circular Linked list

```
//STACK USING CIRCULAR LINKED LIST
funcprot(0)
function[link2]=append(ele,link1)
  link2=list
     (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,,0,0)
     ;
  if(link1(1)(1).add==0)
    link1(1)(1).data=ele;
    link1(1)(1).add=1;
    link1(1)(1).nexadd=1;
    link2(1)=link1(1)(1);
    else
    if(link1(1)(1).nexadd==link1(1)(1).add)
      lin2=link1(1)(1);
      lin2.data=ele;
```

```
      lin2.add=link1(1)(1).add+1;
      link1(1)(1).nexadd=lin2.add;
      lin2.nexadd=link1(1)(1).add;
      link2(1)=link1(1)(1);
      link2(2)=lin2;
    else
      lin2=link1(1)(1);
      i=1;
      while(link1(i)(1).nexadd~=link1(1)(1).add)
        i=i+1;
      end
      j=i;
      lin2.data=ele;
      lin2.add=link1(i).add+1;
      lin2.nexadd=link1(1)(1).add;
      link1(i).nexadd=lin2.add;
      link2(1)=link1(1)(1);
      i=2;
      while(link1(i).nexadd~=lin2.add)
        link2(i)=(link1(i));
        i=i+1;
      end
      link2(i)=link1(i);
      link2(i+1)=lin2;
    end
  end
endfunction
function[link2]=add(ele,pos,link1);
   link2=list(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,,0,0);
   i=1;
   while(i<=pos)
     if(link1(i).nexadd==link1(1)(1).add)
       break;
     else
       i=i+1;
     end
```

```
    end
    if(link1(i).nexadd~=link1(1)(1).add)
      i=i-1;
       lin2.data=ele;
       lin2.add=i;
       j=i;
       while(link1(j).nexadd~=link1(1)(1).add)
         link1(j).add=link1(j).add+1;
         link1(j).nexadd=link1(j).nexadd+1;
         j=j+1;
       end
       link1(j).add=link1(j).add+1;
       lin2.nexadd=link1(i).add;
       link1(i-1).nexadd=lin2.add;
       k=1;
       while(k<i)
          link2(k)=link1(k);
          k=k+1;
        end
        link2(k)=lin2;
        k=k+1;
        link2(k)=link1(k-1);
        k=k+1
        l=k-1;
        while(k~=j)
          link2(k)=link1(l);
          k=k+1;
          l=l+1;
        end
        link2(j)=link1(j-1);;
        link2(j+1)=link1(j);
      else
        if(i==pos)
          k=1;
          lin2.data=ele;
          lin2.add=link1(i-1).add+1;
          link1(i).add=link1.add+1;
          lin2.nexadd=link1(i).add;
```

```
          link1(i).nexadd=link1(1)(1).add;
          k=1;
          while(k<pos)
            link2(k)=link1(k);
            k=k+1;
          end
          link2(k)=lin2;
          link2(k+1)=link1(k)
        end
      end

endfunction
function[link2]=delete1(pos,link1)
  link2=list
     (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,,0,0)
     ;
  i=1;
  if(link1(1)(1).add==0)
    disp("Invalid");
  else
    if(link1(1)(1).nexadd==link1(1)(1).add)
      link1(1)(1).add=0;
      link1(1)(1).nexadd=0;
      link1(1)(1).data=0;
      link2(1)=link1(1)(1);
    else
  while(i<=pos)
    if((link1(i).nexadd==link1(1)(1).add))
      break;
    else
      i=i+1;
    end
  end
  if(link1(i).nexadd~=link1(1)(1).add)
    i=i-1;
    j=1;
    if(i==1)
      j=1;
```

```
      while(link1(j).nexadd~=link1(1)(1).add)
        link2(j)=link1(j);
        j=j+1;
      end
      link2(j)=link1(j);
    else
    link1(i-1).nexadd=link1(i+1).add;
    while(link1(j).nexadd~=link1(i+1).add)
      link2(j)=link1(j);
      j=j+1;
    end
    if(j~=i-1)
      link2(j)=link1(j);
      link2(j+1)=link1(j+1);
      k=i+1;
      l=2;
    else
      link2(j)=link1(j);
      k=i+1;
      l=1;
    end
    while(link1(k).nexadd~=link1(1)(1).add)
      link2(j+l)=link1(k);
      k=k+1;
      l=l+1;
    end
    link2(j+l)=link1(k);
  end
  else
    if(i==pos)
      j=1;
      link1(i-1).nexadd=link1(1)(1).add;
      while(j<=i-1)
        link2(j)=link1(j);
        j=j+1;
      end
    end
  end
```

```
end
end

endfunction
function[sta]=push(ele,stack)
  if(stack.top==0)
    stack.a=ele;
    stack.top=stack.top+1;
    sta=stack;
  else
    i=1;
    link1=struct('data',0,'add',0,'nexadd',0);
    while(i<=stack.top)
      link1=append(stack.a(i),link1);
      i=i+1;
    end
    link1=append(ele,link1);
    stack.top=stack.top+1;
    a=[stack.a(:,:) link1(stack.top).data];
    stack.a=a;
    sta=stack;
  end
endfunction
function[ele,sta]=pop(stack)
  ele=-1;
  sta=0;
  if(stack.top==0)
    disp("Stack Underflow");
    return;
else
   i=1;
    link1=struct('data',0,'add',0,'nexadd',0);
    while(i<=stack.top)
      link1=append(stack.a(i),link1);
      i=i+1;
    end
    ele=link1(stack.top).data;
    link1=delete1(stack.top,link1);
```

```
    stack.top=stack.top-1;
    i=2;
    a=link1(1)(1).data
    while(i<=stack.top)
      a=[a(:,:) link1(i).data];
      i=i+1;
    end
    stack.a=a;
    sta=stack;
  end
endfunction
function[stack]=empty()
  stack=struct('a',0,'top',0);
endfunction
//Calling Routine:
stack=empty()//Create an empty stack
stack=push(4,stack);
stack=push(55,stack);
stack=push(199,stack);
stack=push(363,stack);
[ele,stack]=pop(stack);
disp(stack,"After the above operations stack is:");
```

Scilab code Exa 4.6 Implementing Priority Queue Using Lists

```
//Implementing Priority Queue Using Lists
funcprot(0)
function[link2]=insert_pri(ele,link1)
  link2=list
     (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,,0,0)
     ;
  if(link1(1)(1).add==0)
    link1(1)(1).data=ele;
    link1(1)(1).add=1;
    link1(1)(1).nexadd=1;
```

```
    link2(1)=link1(1)(1);
  else
    if(link1(1)(1).nexadd==link1(1)(1).add)
      if(ele>=link1(1)(1).data)
        t=ele;
        p=link1(1)(1).data;
      else
        t=link1(1)(1).data;
        p=ele;
      end
      link1(1)(1).data=t;
      lin2=link1(1)(1);
      lin2.data=p;
      lin2.add=2;
      lin2.nexadd=link1(1)(1).add;
      link1(1)(1).nexadd=lin2.add;
      link2(1)=link1(1)(1);
      link2(2)=lin2;
    else
      i=1;
      a=[];
      while(link1(i).nexadd~=link1(1)(1).add)
        a=[a(:,:) link1(i).data];
        i=i+1;
      end
      a=[a(:,:) link1(i).data];
      a=gsort(a);
      j=1;
      while(j<=i)
        link1(j).data=a(j);
        j=j+1;
      end
      k=1;
      while(link1(k).data>=ele)
        if(link1(k).nexadd==link1(1)(1).add)
          break;
        else
          link2(k)=link1(k);
```

```
            k=k+1;
          end
        end
         if(link1(k).nexadd~=link1(1)(1).add)
           lin2=link1(k);
           lin2.data=ele;
           lin2.add=link1(k).add;
           j=k;
           y=link1(1)(1).add;
           while(link1(k).nexadd~=y)
             link1(k).add=link1(k).add+1;
             link1(k).nexadd=link1(k).nexadd+1;
             k=k+1;
           end
           link1(k).add=link1(k).add+1;
           lin2.nexadd=link1(j).add;
           link2(j)=lin2;
           j=j+1;
           while(j<=k+1)
             link2(j)=link1(j-1);
             j=j+1;
           end
         else
           lin2=link1(k);
           lin2.data=ele;
           lin2.nexadd=link1(1)(1).add;
           lin2.add=link1(k).add+1;
           link1(k).nexadd=lin2.add;
           j=1;
           while(j<=k)
             link2(j)=link1(j);
             j=j+1;
           end
           link2(j)=lin2;
         end
       end
     end
endfunction
```

```
function[ele,link2]=extract_min(link1);
  link2=list
     (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,,0,0)
     ;
  i=1;
  ele=-1;
  if(link1(1)(1).add==0)
    disp("Underflow");
    return;
  else
    if(link1(1)(1).nexadd==link1(1)(1).add)
      link1(1)(1).add=0;
      link1(1)(1).nexadd=0;
      ele=link1(1)(1).data;
      link1(1)(1).data=0;
      link2(1)=link1(1)(1);
    else
      i=1;
      while(link1(i).nexadd~=link1(1)(1).add)
        link2(i)=link1(i);
        i=i+1;
      end
      ele=link1(i).data;
      link2(i-1).nexadd=link2(1).add;
    end
  end
endfunction
//Calling Routine:
link1=struct('data',0,'add',0,'nexadd',0);
link1=insert_pri(3,link1);
link1=insert_pri(4,link1);
link1=insert_pri(22,link1);
link1=insert_pri(21,link1);
link1=insert_pri(11,link1);
disp(link1,"List After Insertions");
[ele,link1]=extract_min(link1)
disp(ele,"Element after the min extraction");
```

Chapter 5

Trees

Scilab code Exa 5.1 Implementing Binary Tree

```

funcprot(0);
function[tree]=maketree(x)
  tree=zeros(30,1);
  for i=1:30
    tree(i)=-1;
  end
  tree(1)=x;
  tree(2)=-2;
endfunction
function[tree1]=setleft(tree,tre,x)
  tree1=[];
  i=1;
  while(tree(i)~=-2)
    if(tree(i)==tre)
      j=i;
    end
    i=i+1;
  end
  if(i>2*j)
    tree(2*j)=x;
```

```
  else
    tree(2*j)=x;
    tree(2*j+1)=-2;
    for l=i:2*j-1
      tree(i)=-1;
    end
  end
  tree1=tree;
endfunction
function[tree1]=setright(tree,tre,x)
  tree1=[];
  i=1;
  while(tree(i)~=-2)
    if(tree(i)==tre)
      j=i;
    end
    i=i+1;
  end
  if(i>2*j+1)
    tree(2*j+1)=x;
  else
    tree(2*j+1)=x;
    tree(2*j+2)=-2;
    for l=i:2*j
      tree(i)=-1;
    end
  end
  tree1=tree;
endfunction
function[x]=isleft(tree,tre)
  i=1;
  x=0;
  while(tree(i)~=-2)
    if(tree(i)==tre)
      j=i;
    end
    i=i+1;
  end
```

```
  if(i>=2*j)
    if((tree(2*j)~=-1)|(tree(2*j)~=-2))
      x=1;
      return 1;
    else
      return 0;
    end
  else
    x=0;
    return x;
  end
endfunction
function[x]=isright(tree,tre)
  i=1;
  x=0;
  while(tree(i)~=-2)
    if(tree(i)==tre)
      j=i;
    end
    i=i+1;
  end
  if(i>=2*j+1)
    if((tree(2*j+1)~=-1)|(tree(2*j+1)~=-2))
      x=1;
      return 1;
    else
      return 0;
    end
  else
    x=0;
    return x;
  end
endfunction
//Calling Routine:
tree=maketree(3);
disp(tree,"Tree made");
tree=setleft(tree,3,1);
disp(tree,"After setting 1 to left of 3");
```

```
tree=setright(tree,3,2);
disp(tree,"After setting 2 to right of 3");
tree=setright(tree,2,4);
tree=setleft(tree,2,5);
tree=setright(tree,1,6);
tree=setright(tree,5,8);
disp(tree,"After above operations:");
x=isright(tree,3);
disp(x,"Checking for the right son of 3 yes if 1 else no");
x=isleft(tree,2);
disp(x,"Check for left son of 2");
```

Scilab code Exa 5.2 Tree Trversal Techniques

```
funcprot(0);
function[tree]=maketree(x)
  tree=zeros(30,1);
  for i=1:30
    tree(i)=-1;
  end
  tree(1)=x;
  tree(2)=-2;
endfunction
function[tree1]=setleft(tree,tre,x)
  tree1=[];
  i=1;
  while(tree(i)~=-2)
    if(tree(i)==tre)
      j=i;
    end
    i=i+1;
  end
  if(i>2*j)
    tree(2*j)=x;
```

```
  else
    tree(2*j)=x;
    tree(2*j+1)=-2;
    for l=i:2*j-1
      tree(i)=-1;
    end
  end
  tree1=tree;
endfunction
function[tree1]=setright(tree,tre,x)
  tree1=[];
  i=1;
  while(tree(i)~=-2)
    if(tree(i)==tre)
      j=i;
    end
    i=i+1;
  end
  if(i>2*j+1)
    tree(2*j+1)=x;
  else
    tree(2*j+1)=x;
    tree(2*j+2)=-2;
    for l=i:2*j
      tree(i)=-1;
    end
  end
  tree1=tree;
endfunction
function[x]=isleft(tree,tre)
  i=1;
  x=0;
  while(tree(i)~=-2)
    if(tree(i)==tre)
      j=i;
    end
    i=i+1;
  end
```

```
  if(i>=2*j)
    if((tree(2*j)~=-1)|(tree(2*j)~=-2))
      x=1;
      return 1;
    else
      return 0;
    end
  else
    x=0;
    return x;
  end
endfunction
function[x]=isright(tree,tre)
  i=1;
  x=0;
  while(tree(i)~=-2)
    if(tree(i)==tre)
      j=i;
    end
    i=i+1;
  end
  if(i>=2*j+1)
    if((tree(2*j+1)~=-1)|(tree(2*j+1)~=-2))
      x=1;
      return 1;
    else
      return 0;
    end
  else
    x=0;
    return x;
  end
endfunction
funcprot(0);
function[]=inorder(tree,p)
  if(tree(p)==-1|tree(p)==-2)
    return;
  else
```

```
    inorder(tree,2*p);
    printf("%d\t",tree(p));
    inorder(tree,2*p+1);
  end
endfunction
function []=preorder(tree,p)
  if(tree(p)==-1|tree(p)==-2)
    return;
  else
    printf("%d\t",tree(p));
    preorder(tree,2*p);
    preorder(tree,2*p+1);
  end
endfunction
function []=postorder(tree,p)
  if(tree(p)==-1|tree(p)==-2)
    return;
  else
    postorder(tree,2*p);
    postorder(tree,2*p+1);
    printf("%d\t",tree(p));
  end
endfunction
//Calling Routine:
tree=maketree(3);
tree=setleft(tree,3,1);
tree=setright(tree,3,2);
tree=setleft(tree,2,4);
tree=setright(tree,2,5);
disp("Inorder traversal");
inorder(tree,1);
disp("Preorder traversal");
preorder(tree,1);
disp("Postorder traversal");
postorder(tree,1);
```

Scilab code Exa 5.3 Implementing And traversing a Binary Search Tree

```
funcprot(0);
function[tree]=maketree(x)
  tree=zeros(1,30);
  for i=1:30
    tree(i)=-1;
  end
  tree(1)=x;
  tree(2)=-2;
endfunction
function[tree1]=setleft(tree,tre,x)
  tree1=[];
  i=1;
  while(tree(i)~=-2)
    if(tree(i)==tre)
      j=i;
    end
    i=i+1;
  end
  if(i>2*j)
    tree(2*j)=x;
  else
    tree(2*j)=x;
    tree(2*j+1)=-2;
    for l=i:2*j-1
      tree(i)=-1;
    end
  end
  tree1=tree;
endfunction
function[tree1]=setright(tree,tre,x)
  tree1=[];
  i=1;
```

```
  while(tree(i)~=-2)
    if(tree(i)==tre)
      j=i;
    end
    i=i+1;
  end
  if(i>2*j+1)
    tree(2*j+1)=x;
  else
    tree(2*j+1)=x;
    tree(2*j+2)=-2;
    for l=i:2*j
      tree(i)=-1;
    end
  end
  tree1=tree;
endfunction
function[x]=isleft(tree,tre)
  i=1;
  x=0;
  while(tree(i)~=-2)
    if(tree(i)==tre)
      j=i;
    end
    i=i+1;
  end
  if(i>=2*j)
    if((tree(2*j)~=-1)|(tree(2*j)~=-2))
      x=1;
      return 1;
    else
      return 0;
    end
  else
    x=0;
    return x;
  end
endfunction
```

```
function [x]=isright(tree,tre)
  i=1;
  x=0;
  while(tree(i)~=-2)
    if(tree(i)==tre)
      j=i;
    end
    i=i+1;
  end
  if(i>=2*j+1)
    if((tree(2*j+1)~=-1)|(tree(2*j+1)~=-2))
      x=1;
      return 1;
    else
      return 0;
    end
  else
    x=0;
    return x;
  end
endfunction
funcprot(0);
function []=inorder(tree,p)
  if(tree(p)==-1|tree(p)==-2)
    return;
  else
    inorder(tree,2*p);
    disp(tree(p),"  ");
    inorder(tree,2*p+1);
  end
endfunction
function []=preorder(tree,p)
  if(tree(p)==-1|tree(p)==-2)
    return;
  else
    disp(tree(p),"  ");
    preorder(tree,2*p);
    preorder(tree,2*p+1);
```

```
  end
endfunction
function []=postorder(tree,p)
  if(tree(p)==-1|tree(p)==-2)
    return;
  else
    postorder(tree,2*p);
    postorder(tree,2*p+1);
    disp(tree(p),"  ");
  end
endfunction
function [tree1]=binary(tree,x)
  p=1;
  while(tree(p)~=-1&tree(p)~=-2)
    q=p;
    if(tree(p)>x)
      p=2*p;
    else
      p=2*p+1;
    end
  end
  i=1;
  while(tree(i)~=-2)
    i=i+1;
  end
  if(tree(q)>x)
    if(i==2*q)
      tree(2*q)=x;
      tree(2*q+1)=-2
    else
      if(i<2*q)
        tree(i)=-1;
        tree(2*q+1)=-2;
        tree(2*q)=x;
      end
    end

  else
```

```
      if(i==2*q+1)
        tree(2*q+1)=x;
        tree(2*q+2)=-2;
      else
        if(i<2*q+1)
          tree(i)=-1;
          tree(2*q+1)=x;
          tree(2*q+2)=-2;
        end
      end

    end
    tree1=tree;
endfunction
//Calling Routine:
tree=maketree(3);
tree=binary(tree,1);
tree=binary(tree,2);
tree=binary(tree,4);
tree=binary(tree,5);
disp(tree,"Binary tree thus obtaine by inserting
    1,2,4and5 in tree rooted 3 is:");
```

Scilab code Exa 5.4 Checking the duplicate number using BST

```
function[tree1]=binary(tree,x)
  p=1;
  while(tree(p)~=-1&tree(p)~=-2)
    q=p;
    if(tree(p)>x)
      p=2*p;
    else
      p=2*p+1;
    end
  end
```

```
  if(tree(q)>x)
    if(tree(2*q)==-2)
      tree(2*q)=x;
      tree(2*q+1)=-2;
    else
      tree(2*q)=x;
    end
  else
    if(tree(2*q+1)==-2)
      tree(2*q+1)=x;
      tree(2*q+2)=-2;
    else
      tree(2*q+1)=x;
    end
  end
  tree1=tree;
endfunction
funcprot(0);
function[tree]=maketree(x)
  tree=zeros(40,1);
  for i=1:40
    tree(i)=-1;
  end
  tree(1)=x;
  tree(2)=-2;
endfunction
function[]=duplicate1(a,n)
  tree=maketree(a(1));
  q=1;
  p=1;
  i=2;
  x=a(i)
  while(i<n)
    while(tree(p)~=x&tree(q)~=-1&tree(q)~=-2)
      p=q;
      if(tree(p)<x)
        q=2*p;
      else
```

```
        q=2*p+1;
      end
    end
    if(tree(p)==x)
      disp(x," Duplicate  ");
    else
      tree=binary(tree,x);
    end
    i=i+1;
    x=a(i);
  end
  while(tree(p)~=x&tree(q)~=-1&tree(q)~=-2)
      p=q;
      if(tree(p)<x)
        q=2*p;
      else
        q=2*p+1;
      end
    end
    if(tree(p)==x)
      disp(x," Duplicate  ");
    else
      tree=binary(tree,x);
    end
endfunction
//Calling Adress:
a=[22 11 33 22 211 334]
duplicate1(a,6)
a=[21 11 33 22 22 334]
duplicate1(a,6)
```

Chapter 6

Sorting

Scilab code Exa 6.1 Bubble Sort

```
function[a1]=bubble(a,n)
  i=1;
  j=1;
  temp=0;
  for i=1:n-1
   for j=1:n-i
      if(a(j)>a(j+1))
        temp=a(j);
        a(j)=a(j+1);
        a(j+1)=temp;
      end
      j=j+1;
    end
    i=i+1;
  end
  a1=a;
  disp(a1,"Sorted array is:");
endfunction
//Calling Routine:
a=[23 21 232 121 2324 1222433 1212]
disp(a,"Given Array");
```

```
a1=bubble(a,7)
```

Scilab code Exa 6.2 Quick Sort

```
function[a1]=quick(a);
  a=gsort(a);//IN BUILT QUICK SORT FUNCTION
  n=length(a);
  a1=[];
  for i=1:n
    a1=[a1(:,:) a(n+1-i)];
  end
  disp(a1,"Sorted array is:");
endfunction
//Calling Routine:
a=[23 21 232 121 2324 1222433 1212]
disp(a,"Given Array");
a1=quick(a)
```

Scilab code Exa 6.3 Selection Sort

```
function[a1]=selection(a,n)
  i=n;
  while(i>=1)
    large=a(1);
    indx=1;
    for j=1:i
      if(a(j)>large)
        large=a(j);
        indx=j;
      end
    end
    a(indx)=a(i);
    a(i)=large;
```

```
    i=i-1;
  end
  a1=a;
  disp(a1,"Sorted array is:");
endfunction
//Calling Routine:
a=[23 21 232 121 2324 1222433 1212]
disp(a,"Given Array");
a1=selection(a,7)
```

Scilab code Exa 6.4 Insertion Sort

```
function[a1]=insertion(a,n)
  for k=1:n
    y=a(k);
    i=k;
    while(i>=1)
      if(y<a(i))
        a(i+1)=a(i);
        a(i)=y;
      end
      i=i-1;
    end
  end
  a1=a;
  disp(a1,"Sorted array is:");
endfunction
//Calling Routine:
a=[23 21 232 121 2324 1222433 1212]
disp(a,"Given Array");
a1=insertion(a,7)
```

Scilab code Exa 6.5 Shell sort

```
function[a1]=shell(a,n,incr,nic)
  for i=1:nic
    span=incr(i);
    for j=span+1:n
      y=a(j);
      k=j-span;
      while(k>=1&y<a(k))
          a(k+span)=a(k);
        k=k-span;
      end
      a(k+span)=y;
    end
  end
  a1=a;
  disp(a1,"Sorted array is:");
endfunction
//Calling Routine:
a=[23 21 232 121 2324 1222433 1212]
disp(a,"Given Array");
incr=[5 3 1]//must always end with 1
a1=shell(a,7,incr,3)
```

Scilab code Exa 6.6 Merge Sort

```
function[a1]=mergesort(a,p,r)
  if(p<r)
    q=int((p+r)/2);
    a=mergesort(a,p,q);
    a=mergesort(a,q+1,r);
    a=merge(a,p,q,r);
  else
    a1=a;
    return;
  end
  a1=a;
```

```
endfunction
function[a1]=merge(a,p,q,r)
  n1=q-p+1;
  n2=r-q;
  left=zeros(n1+1);
  right=zeros(n2+1);
  for i=1:n1
    left(i)=a(p+i-1);
  end
  for i1=1:n2
      right(i1)=a(q+i1);
  end
  left(n1+1)=999999999;
  right(n2+1)=999999999;
  i=1;
  j=1;
  k=p;
  for  k=p:r
    if(left(i)<=right(j))
      a(k)=left(i);
      i=i+1;
    else
      a(k)=right(j);
      j=j+1;
    end
  end
  a1=a;
endfunction
//Calling Routine:
a=[23 21 232 121 26324 1222433 14212]
disp(a,"Given Array");
a1=mergesort(a,1,7)
disp(a1,"Sorted array is:");
a=[232 11212 3443 23221 123424 32334 12212 2443 232]
disp(a,"Given Array");
a1=mergesort(a,1,9);
disp(a1,"Sorted Array");
```

Scilab code Exa 6.7 Binary Tree Sort

```
function[tree1]=binary(tree,x)
  p=1;
  while(tree(p)~=-1&tree(p)~=-2)
    q=p;
    if(tree(p)>x)
      p=2*p;
    else
      p=2*p+1;
    end
  end
  if(tree(q)>x)
      tree(2*q)=x;
  else
      tree(2*q+1)=x;
  end
  tree1=tree;
endfunction
funcprot(0);
function[tree]=maketree(x)
  tree=zeros(100,1);
  for i=1:100
    tree(i)=-1;
  end
  tree(1)=x;
  tree(2)=-2;
endfunction
function[]=inorder(tree,p)
  if(tree(p)==-1|tree(p)==-2)
    return;
  else
    inorder(tree,2*p);
    printf("%d\t",tree(p));
```

```
    inorder(tree,2*p+1);
  end
endfunction
function[]=binsort(a,n)
  a1=maketree(a(1))
  for i=2:n
    a1=binary(a1,a(i));
  end
  disp("Sorted array is:");
  inorder(a1,1);
endfunction
//Calling Routine:
a=[23 21 232 121 2324 1222433 1212]
disp(a,"Given Array");
a1=binsort(a,7)
```

Chapter 7

Searching

Scilab code Exa 7.1 Sequential Search

```
function[]=search(a,n,ele)
  i=1;
  j=0;
  for i=1:n
    if(a(i)==ele)
      printf("Found %d AT %d\n",ele,i);
      j=1;
    end
  end
  if(j==0)
    disp("%d NOT FOUND",ele);
  end
endfunction
//Calling Routine:
a=[2 33 22 121 23 233 222]
disp(a,"Given array");
search(a,7,23)
```

Scilab code Exa 7.2 Sorted sequential search

```
function []=sortedsearch(a,n,ele)
  if(a(1)>ele|a(n)<ele)
    disp("NOT IN THE LIST");
  else
    i=1;
    j=0;
    for i=1:n
      if(a(i)==ele)
        printf("FOUND %d AT %d",ele,i);
        j=1;
      else
        if(a(i)>ele)
          break;
        end
      end
    end
    if(j==0)
      disp("%d NOT FOUND",ele);
    end
  end
endfunction
//Calling Routine:
a=[2 22 23 33 121 222 233]//a should be  sorted
disp(a,"Given array");
search(a,7,23)
```

Scilab code Exa 7.3 Binary Search

```
function []=binsearch(a,n,i)
  l=1;
  h=n;
  while(l<=h)
    mid=int((l+h)/2);
    if(a(mid)==i)
      printf("FOUND %d AT %d",i,mid);
```

```
      break;
    else
      if(a(mid)>i)
        h=mid-1;
      else
        l=mid+1;
      end
    end
  end
endfunction
//Calling Routine:
a=[2 22 23 33 121 222 233]//a should be  sorted
disp(a,"Given array");
search(a,7,23)
```

Chapter 8

Graphs

Scilab code Exa 8.1 Simple Graph Functions

```
//Simple Graph Functions
function[]=graph();

  i=1,j=1;
  adj=zeros(10000);
  for i=1:n
  for j=1:n

      adj((i-1)*n+j)=temp;
    end
  end
  for i=1:n
    for  j=1:n
      if((adj((i-1)*n+j))==1)
        printf("Vertex %d is connected to vertex %d\
          n",i,j);
      end
    end
  end

endfunction
```

Scilab code Exa 8.2 Finding The Number Of Paths From One Vertex-ToOther

```
//Finding The Number Of Paths From One Vertex To
   Another Of A Given Length

function[b]=path(k,n,adj,i,j)
  b=0;
  if(k==1)
    b=adj((i-1)*n+j);
  else
    for c=1:n
      if(adj((i-1)*n+c)==1)
        b=b+path(k-1,n,adj,c,j);
      end
    end
  end
    printf("Number of paths from vertex %d to %d of
       length %d are %d",i,j,k,b);
  return b;
endfunction
//Calling Routine:
n=3;
adj=[0 1 1 0 0 1 0 0 0]
b=path(1,n,adj,1,3)
```

Scilab code Exa 8.3 Finding The Number Of Simple Paths From One Point

```
//Finding The  Number Of Simple Paths From One Point
    To Another In A Given Graph
```

```
funcprot(0)
function[]=sim_path(n,adj,i,j);
  l=0;
  m=1;
  for m=1:n
    l=l+path(m,n,adj,i,j);
  end
  printf("There are %d  simple paths from %d to %d
     in the given graph\n",l,i,j);
endfunction
function[b]=path(k,n,adj,i,j)
  b=0;
  if(k==1)
    b=adj((i-1)*n+j);
  else
    for c=1:n
      if(adj((i-1)*n+c)==1)
        b=b+path(k-1,n,adj,c,j);
      end
    end
  end
  return b;
endfunction
n=3;
adj=[0 1 1 0 0 1 0 0 0];
b=sim_path(n,adj,1,3)
```

Scilab code Exa 8.4 Finding Transitive Closure

```
//Finnding Transitive Closure
funcprot(0)
function[path]=Tranclose(adj,n);
  i=1,j=1;
  path=zeros(n*n,1);
  path=tranclose(adj,n);
```

```
  printf("Transitive Closure Of Given Graph is:\n");
  for i=1:n
    printf("For Vertex %d\n",i);
    for j=1:n
      printf(" %d %d is %d\n",i,j,path((i-1)*n+j));
    end
  end

endfunction
function[path]=tranclose(adj,n)
  adjprod=zeros(n*n,1);
  k=1;
  newprod=zeros(n*n,1);
  for i=1:n
    for j=1:n
      path((i-1)*n+j)=adj((i-1)*n+j);
      adjprod((i-1)*n+j)= path((i-1)*n+j);
    end
  end
  for i=1:n
    newprod=prod1(adjprod,adj,n);
    for j=1:n
      for k=1:n
        path((j-1)*n+k)=path((j-1)*n+k)|newprod((j-1)*n+k);
      end
    end
    for j=1:n
      for k=1:n
        adjprod((j-1)*n+k)=newprod((j-1)*n+k);
      end
    end
  end
endfunction
function[c]=prod1(a,b,n)
  for i=1:n
    for j=1:n
      val=0
```

```
      for k=1:n
        val=val|(a((i-1)*n+k)&b((k-1)*n+j));
      end
      c((i-1)*n+j)=val;
    end
  end
endfunction
//Calling Routine:
n=3;
adj=[0 1 0 0 0 1 0 0 0]
path=Tranclose(adj,n)
```

Scilab code Exa 8.5 Warshalls Algorithm

```
//Warshall's Algorithm
funcprot(0)
function[path]=transclose(adj,n)
  for  i=1:n
    for j=1:n
      path((i-1)*n+j)=adj((i-1)*n+j);
    end
  end
  for k=1:n
    for  i=1:n
      if(path((i-1)*n+k)==1)
        for j=1:n
          path((i-1)*n+j)=path((i-1)*n+j)|path((k-1)
             *n+j);
        end
      end
    end
  end
  printf("Transitive closure for the given graph is
     :\n");
  for  i=1:n
```

```
    printf("For vertex %d \n",i);
    for j=1:n
       printf("%d %d is %d\n",i,j,path((i-1)*n+j));
    end
  end
endfunction
//Calling Routine:
n=3;
adj=[0 1 0 0 0 1 0 0 0]
path=Tranclose(adj,n)
```

Scilab code Exa 8.6 Depth First Search Traversal

```
//Depth First Search Traversal
funcprot(0)
function[]=Dfs(adj,n);
  i=1,j=1;
  colour=[];
  for i=1:n
  for j=1:n
      colour=[colour(:,:) 0];
    end
  end
  disp("The DFS traversal is");
dfs(adj,colour,1,n);
endfunction
function[]=dfs(adj,colour,r,n)
  colour(r)=1;
  disp(r,"  ");
  for i=1:n
    if(adj((r-1)*n+i)&(colour(i)==0))
      dfs(adj,colour,i,n);
    end
  end
  colour(r)=2;
```

```
endfunction
//Calling Routine:
n=4;
adj=[0 1 1 0 0 0 0 1 0 0 0 1 0 0 0 0]
Dfs(adj,n)
```

Scilab code Exa 8.7 BFS Traversal

```
////BFS Traversal
funcprot(0)
function[q2]=push(ele,q1)
  if(q1.rear==q1.front)
    q1.a=ele;
    q1.rear=q1.rear+1;
  else
    q1.a=[q1.a(:,:) ele];
    q1.rear=q1.rear+1;
  end
  q2=q1;
endfunction
function[ele,q2]=pop(q1)
  ele=-1;
  q2=0;
  if(q1.rear==q1.front)
        return;
  else
    ele=q1.a(q1.rear-q1.front);
    q1.front=q1.front+1;
    i=1;
    a=q1.a(1);
    for i=2:(q1.rear-q1.front)
      a=[a(:,:) q1.a(i)];
    end
    q1.a=a;
  end
```

```
  q2=q1;
endfunction

function []=Bfs(adj,n);
  i=1,j=1;
  colour=[];
  for i=1:n
  for j=1:n
      colour=[colour(:,:) 0];
    end
  end
  disp("The BFS Traversal is");
bfs(adj,colour,1,n);
endfunction
function []=bfs(adj,colour,s,n)
  colour(s)=1;
  q=struct('rear',0,'front',0,'a',0);
  q=push(s,q);
  while((q.rear)-(q.front)>0)
    [u,q]=pop(q);
    disp(u," ");
    for i=1:n
      if(adj((u-1)*n+i)&(colour(i)==0))
        colour(i)=1;
        q=push(i,q);
      end
    end
    colour(u)=2;
  end
endfunction
//Calling Routine:
n=4;
adj=[0 1 1 0 0 0 0 1 0 0 0 1 0 0 0 0]
Bfs(adj,n)
```

Scilab code Exa 8.8 Dijkstras Algorithm

```
//Dijkstras Algorithm
funcprot(0)
function[l]=short(adj,w,i1,j1,n)
  for i=1:n
    for j=1:n
      if(w((i-1)*n+j)==0)
        w((i-1)*n+j)=9999;
      end
    end
  end

  distance=[];
  perm=[];
  for i=1:n
    distance=[distance(:,:) 99999];
    perm=[perm(:,:) 0];
  end
  perm(i1)=1;
  distance(i1)=0;
  current=i1;
  while(current~=j1)
    smalldist=9999;
    dc=distance(current);
    for i=1:n
      if(perm(i)==0)
        newdist=dc+w((current-1)*n+i);
        if(newdist<distance(i))
          distance(i)=newdist;
        end
        if(distance(i)<smalldist)
          smalldist=distance(i);
          k=i;
        end
      end
    end
    current=k;
```

```
    perm(current)=1;
  end
  l=distance(j1);
  printf("The  shortest path between %d and %d is %d
    ",i1,j1,l);
endfunction
//Calling Routine:
n=3;
adj=[0 1 1 0 0 1 0 0 0]//Adjacency  List
w=[0 12 22 0 0 9 0 0 0]//weight list fill 0 for no
   edge
short(adj,w,1,3,n);
```